Hybrid Distributed Real Time Scheduling Algorithm

A.Prashanth Rao[1] , Dr.A. Govardhan[2] C.Venu Gopal[3]
[1]*Reasearch Scholar,* JNTU College of Engineering*, Hyderabad (Dt), A.P, India,*
+91-9490232922,
adirajupppy@yahoo.com
[2]Professor of CSE & Principal, *JNTU College of Engineering, KarimNagar (Dt), A.P, India,*
+91-9440887733
govardhan_cse@yahoo.co.in
[3][1]*Reasearch Associate* of CSE, Osmania University, Hyderabad, A.P, India.
cherupalli_v@yahoo.com

Abstract: In the design of real time distributed system, the scheduling problem is considered to be nature of NP Hard and has been addressed in the literature. However due growing complexities of real time applications, there is need to find optimal dynamic scheduling algorithm. In this paper, we describe a heuristic hybrid scheduling algorithm which combines both static and dynamic tasks. Initially a processor can be allocated a fixed number of units based on pre-defined tasks which are generated from different sensors and there will certain number of units are meant for dynamically created tasks. When a dynamic task arrives at a node, the local scheduler at that node attempts to guarantee that the task will complete execution before its deadline, on that node. If the attempt fails the scheduler searches the node where task will feasibly scheduled. This type of scheduling performs the best results and scheduling algorithm can be configurable.

Keywords*: Multiprocessor, Critical Instant, Deadline, Utilization Value, Scheduling.*

Abbreviations:

RM: Rate Monotonic;
RMST: Rate Monotonic Small Tasks;
RMGT: Rate Monotonic General Tasks;
RMCT: Rate Monotonic Critical Tasks;
RMEDF: Rate Monotonic Earliest Deadline First;

1. INTRODUCTION

Distributed real time systems are becoming more prevalent in applications such as avionics, process control, chemical industry, and robotics. These applications contain many tasks that has period, execution time, deadline which must be met. Tasks in these applications are typically categorized as critical, essential, and nonessential. Critical tasks are defined as those which must meet their deadline under all circumstances; otherwise it leads to a catastrophe. Essential tasks are those that have a deadline and are important in the operation of the system, but will not cause a catastrophe if they are finished on time. If essential tasks miss their deadline, the performance of the system degrades. Nonessential tasks are defined are those whose deadline if missed will not affect the system in the near future but may have effect in the long run. Maintenance and Bookkeeping activities fall in this category. All these types of tasks are to be scheduled without missing deadline.

A scheduling algorithm provides a set of rules that determine the processor(s) or node to be used and tasks to be executed at any particular point of time. There are many scheduling algorithms to handle fixed priority tasks are known as static scheduling algorithms [1], [2], [3]. However, static scheduling for task groups determines a poor effiencey in resource usage, which could become unacceptable in many applications scenarios. Similarly there are many dynamic scheduling algorithms [4], [5], [6] of group of tasks which uses Earliest Deadline First (EDF) algorithm. There are several schedulability conditions for RM algorithm which are predominantly oriented towards task periods. Two off line algorithms namely RMST [8] and RMGT [8] which were developed, order the tasks according to their periods and scheduling.

Most of real applications uses both algorithms and there need to combine these two algorithms to schedule the real time tasks. In this paper we need to develop a hybrid algorithm which is used to schedule both periodic and non-periodic tasks. So we should have a proper mix of low and high priority tasks and also separate set are formed.

This paper is organized as follows. Section 2 describes Basic Terminology. System Model in Section 3. Section 4 describes global scheduling Results and Discussions in Sections 5. Finally, the conclusion and future scope are given in Section 6.

2. BASIC TERMINOLOGY

The real time tasks which were generated from different sources such as sensors, which are treated as static tasks and are repeated in regular interval of time. These tasks in turn to create other tasks known as aperiodic tasks and arrival of these are not known in advance. Some other

DOI: 10.5176_2010-2283_1.2.54

tasks also generated due environmental change in the system. These types of tasks are known as dynamic in nature. Tasks in real time systems are periodic and aperiodic.

a) **Periodic tasks:** Periodic tasks are time driven and recur at regular interval of time called the period. Generally sensor generates periodically these types of the tasks. These tasks are known as static in nature and characterized by periodicity and worst case execution time. These tasks are scheduled using RM algorithm.

b) **Aperiodic tasks:** Aperiodic tasks are event-driven and activated only when certain events occur. Aperiodic task whose inter-arrival times are known as **sporadic tasks**. These types of tasks are dynamic in nature and characterized by phase (ǿ), worst case execution time (e_i) and deadline (d_i). These tasks are scheduled using EDF algorithm.

Both type of tasks such as periodic and aperiodic will together scheduled when they arrive.

c) **Laxity:** The Laxity of $Task_i$ is defined to be (d_i – e_i - t) where d_i is the deadline , e_i execution time and t is current time

d) **Maximum Periodicity (P_{max}):** If P_{max} is the maximum periodicity constraint, then for a feasible system T< P_{max} . Where T is the total execution for a system task.

Using above definitions we can develop a scheduling algorithms which we discussed in detailed in the section (4), before that we need understand system model which describes the following section.

3. SYSTEM MODEL

In this section we describe certain characteristics of system and these are used to develop the proposed algorithm.

1. Tasks are periodic in nature and it is characterized by periodicity (p_i), execution time (e_i) and arrival times are known in advance. Fixed priority scheduling algorithm used to schedule these tasks.
2. Tasks are aperiodic, i.e. the task arrivals are not known a priori. Every $task_i$ characterized by arrival time (a_i), ready time (r_i),worst case computation time (e_i) and deadline (d_i)
3. Periodic tasks are preemptable where as aperiodic tasks are non-preemptable
4. There are m nodes in loosely coupled distributed system and each node allocate fixed number of execution units which equal to half of its time critical value (M) or P_{max} and rest of half units are meant for non-periodic tasks. This means time axis is divided into two partitions windows in which one window statically pre-assigned tasks and other window dynamically allocated tasks.
5. The first window uses RM algorithm where as other window uses EDF algorithm.
6. M denotes the least common multiples of p_i , ∀ j=1... n. A time interval of length M is called a hyper-period or Critical Time Value of the tasks which are allocated to that particular processor.
7. Critical Time Interval is the time gap or time interval between two consecutive instants of time when at those instants of all the tasks arrive simultaneously.

   If all tasks arrive at time t=0 and again the next instant of arrival of tasks is at time t=M. The interval [0, M] is Critical Time Interval.

   On each node there are three components involved in the system. The local scheduler who is used to schedules the task at given node. The dispatcher invokes the next task to be executed and global scheduler schedules the task if it is not schedule locally. These concepts are pre-request to understand global scheduling algorithm.

4. GLOBAL SCHEDULING

There are m nodes in a loosely coupled distributed system and each node contains set of resources such as processor. A task is an execution entity and its execution can be preempted if it is scheduled based on RM algorithm otherwise non-preemptive. Before presenting scheduling algorithm we need to define task system and its feasibility.

4.1.TASK SYSTEM AND FEASIBITY

Typically the real time systems with computer oriented control of a process involves execution of independent task system $\boldsymbol{\Gamma} = \{\tau_1, \tau_2, \tau_3, \tau_4 \ldots \tau_n\}$, each task $\boldsymbol{\tau_j} \in \boldsymbol{\Gamma}$ has p_i period and an execution time e_j. In reality and in general, the total execution period (P_{max}) of $\boldsymbol{\Gamma}$ for task $\boldsymbol{\tau_j}$ $\in$ Γ, $P_{max} \neq \sum_{j=1}^{n} e_j$. It is quite reasonable to state that $P_{max} \geq \sum_{j=1}^{n} e_j$ as the periodicity of each task $\boldsymbol{\tau_j} \in \boldsymbol{\Gamma}$ varies from another. We define the periodicity of each task $\boldsymbol{\tau_j}$ as $\mathbf{p_j}$, and it is obvious that the execution time of $\boldsymbol{\tau_j}$ has to be less than that of its periodicity or $e_j < p_j$ otherwise it cannot

be processed completely. The task system can be defined by equation (1) arranging all tasks with increasing periods.

$$\Gamma^1 = \{(p_{i1}, e_{i1}), (p_{i2}, e_{i2}) \ldots (p_{in}, e_{in})\} \text{ ---------- (1)}$$

The above task system (1) is used to schedule the task system for fixed priority algorithm. But each such system generates aperiodic tasks which need to schedule using dynamic priority algorithm. Initially task system partitioned into two or more subsets and each subset will execute on a different processor. The division of subset based on total execution units of periodic task system.

The total executions time (T) units of periodic task system can be estimated and feasibility of tasks is determined. These execution units (T) can be divided into m sets and each set contain the execution time units less than or equal to half of its time critical value (M) or P_{max}. Remaining half of units reserved for non-periodic tasks, their arrival times are not known in advance. On each processor the time axis is divided into two parts, the first is known as static and other window known as dynamic window. The static window uses RM algorithm and dynamic uses EDF algorithm. So we can improve the utilization of the processor. The fixed allocation algorithm uses window which describes below.

4.2 FIXED ALLOCATION ALOGRITHM

Initially tasks are allocated to the processor based on the periodicity (P_{max}) of the task which will be the first element in the queue. The total execution units are set to be P_{max} and these execution units on time axis t will be divided into two windows using configurable parameter k. The configurable parameter value in between 0 and 1 which depends upon the application and its data values. For some applications planning cycle i.e. LCM of periodicity values as taken as P_{max} taken into consideration.

According to RMEDF Algorithm 1(shown below), which divides time axis into two windows one for static allocation and other for dynamic allocation and the number of tasks allocated processor based on K* P_{max} by varying the value of k, we can mix-up the periodic and non-periodic tasks. This algorithm also allocates the fixed periodic tasks to each processor.

When k=1, the window completely loaded with periodic task sets. The RMEDF algorithm converted to RMCT (see Figure1)

The RMEDF Algorithm is used to divide the window into parts and allocates periodic tasks as well free slots are used for dynamically created tasks. The algorithm presented below.

Input: Queue which contains tasks (n) starting from largest periodicity to smallest periodicity

Output: No. of processors required (m_2) and each set divides into two windows.

I.M=LCM(p_1, p_2, p_3, p_4,..... P_n) ;
// M is the critical instant of periodic task system.

II.I = Set of jobs or number of occurrences in given M of a given task system
i.e $I_1 = M/ p_1$, $I_2 = M/ p_2$,.....$I_n = M/p_n$

$I = \{I_1, I_2, I_3, \ldots\ldots\ldots I_n\}$

III.T = Total execution time units of given task system within the critical instant.

$$T = \sum_{i}^{n} I_i * e_i$$

If T < M than schedule on uniprocessor system i.e. m= 1 go to step VI

I. Compute Number of processors.

i. m_2=0, T=0,i=0, j=n-1;
//Initializing the variables

ii. While((i<n)&&(i<j))
{ T= e_i , temp= P_i;

While (j≥0)

{

T= [temp/p_j]e_j +T;
if (T< k*(temp))
then j—
// k is configurable parameter.
else m_2= m_2+1;}
i++; }

IV.Write the value of m and corresponding m sets, each set contains a set of tasks which are feasible to schedule on given processor.

RMEDF Algorithm 1

4.3 DYNAMIC ALLOCATION ALGORITHM

Assume that each node may contain a set of processors and they need to communicate with each other. When a task arrives at node N_i the local scheduler of N_i is invoked to try to guarantee the newly arrived task on that node. If it is feasible on that node it will add to its local scheduler otherwise it searches where it should be scheduled.

Each node maintains a status table which contains process details, total number of execution units, surplus computational capacity, taskId, and cooperation with other processors. If any processor reaches to maximum utilization for allocating one task, no other task is allocated to that processor. This processor treated as non-cooperative with other processor. This means that processor can not be allocated to any other tasks except aperiodic tasks results from the same periodic task which is generated.

The RMEDF Algorithm 1 allocates periodic tasks to the processor and divides the window into two parts. The dynamic tasks such aperiodic or sporadic tasks..etc, are scheduled using Focused Addressing with Bidding (FAB) [7].The different modules within the FAB are

a) Information policy: Each node in distributed system maintains status table. The status table contains the information about pre-assigned tasks and free available slots which can accommodate the dynamic tasks.

b) Transfer Policy and Selection Policies: When dynamic task arrives at a node, the local scheduler of that node try to schedule the task locally without disturbing pre-assigned tasks. If local scheduler fails, it selects task need to transfer to another light node to schedule the arrived task.

c) Location Policy: If the transfer policy of a node decides to transfer selected task then location policy identifies the suitable node.

The global scheduler sorts the nodes according to available free slots at each and every instant of time. When event occurs at particular instant of time, global scheduler algorithm is invoked.

I. Input: Read task parameters phase (Φ), execution time (e_i) and deadline (d_i).

II. Output: Allocate newly arrived task one the processor.

III.If newly arrived $task_i$ allocated locally then stop

IV.Compute the transfer time t_{trans} of $task_i$ =deadline – (current time (t) +execution time(e_i)

V.Compute free slots between the current time and deadline of newly arrived $task_i$ for each node.

a. Selects suitable $node_j$
b. Allocate $task_i$ to $node_j$
c. Repeat steps I to V for other dynamic tasks.

Algorithm2: global scheduler

Using global scheduler algorithm we can allocate tasks to the processor and assume that the communication between the nodes have fixed time units. If t_{trans} is less than these units then global scheduler selects remote processor if free slots available to execute task.

5. Results and Discussions

Example: Let us illustrate the above algorithm with an example. In Table 1 we show the parameters for a set of 10 tasks after sorting with increasing priority.

According to RMEDF algorithm total time execution units of given task system can be computed and feasibility of the system can be performed. The execution units for each processor can allocate by using above algorithm. Each time configurable parameter is changed to optimal number of units allocated to individual window.

Task id	1	2	3	4	5	6	7	8	9	10
Pi	400	280	230	150	65	60	45	36	20	7
ei	113	141	138	31	16	19	14	11	3	2

Table1:Example RMEDF

When configurable parameter k=1:

RMEDF Alogrithm1 Computes:

Number of processor (m) = 4, which is more optimal than period oriented algorithm.
Task system divides into 4 subsets.
Subset1: {(400,113) ,(20,3),(7,2)}, Allocated to processor1. LCM of periodic tasks = 2800 units.
Subset2: {(280,141), (36, 11)}. Allocated to processor2. LCM of periodic tasks = 2520 units.
Subset3: {(230,138), (45, 14)}. Allocated to processor3. LCM of periodic tasks = 2070 units.
Subset4: {(150, 31), (65, 16), (60, 19)}, Allocated to processor4. LCM of periodic tasks = 3900 units.

For processor 1: Total execution units=7*113 +140*3+400*2=2011, so maximum utilization of processor in only 72% due periodic task set. Remaining (2800-2011) 789 units may utilize properly for background processes. Similarly we can compute available slots for other processor. The processor status table maintains details about processor such processor ID, allocated tasks, available slots for dynamic tasks.

If configurable parameter k reduces by some value, the number of processor may increase but utilization of the processor increases properly by including non-periodic tasks. This means the proper mix of periodic and non-periodic tasks.

K=0.70

RMEDF Alogrithm1 Computes:

Number of processor (m) = 6,
Task system divides into 4 subsets.
Subset1: {(400,113), (7, 2)}, Allocated to processor1. LCM of periodic tasks = 2800 units.
Subset2: {(280,141), (20, 3)}Allocated to processor2. LCM of periodic tasks = 280 units.
Subset3: {(230,138)}. Allocated to processor3. LCM of periodic tasks = 230 units.
Subset4: {(150, 31), (36, 11)}, Allocated to processor4. LCM of periodic tasks = 2700 units.
Subset5: {(65, 16), (45, 14)}, Allocated to processor5. LCM of periodic tasks = 585 units.
Subset6: {(60, 19)}, Allocated to processor6 LCM of periodic tasks = 60 units.

For processor 1: Total execution units=7*113 +400*2=1591. So maximum utilization of processor in only 56% due periodic task set. Remaining (2800-1591) 1209 units may non-periodic tasks which are uses EDF algorithm. Similarly we can compute available slots for other processor.

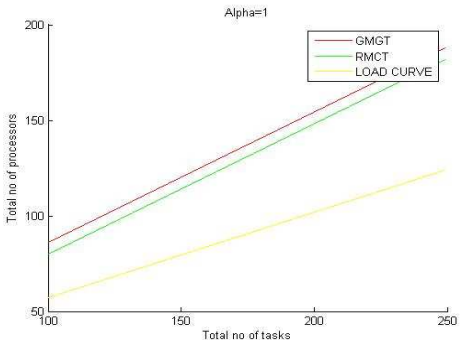
When dynamic task arrived at processor 1 with a phase 20, execution time 1.5 units and deadline 4.

The absolute deadline of the task = 20+4=24.

Global scheduler called when event generates, Scheduler determines that the task not be scheduled locally because high priority task (7, 2) releases at t= 21 and should complete before 23.

Scheduler computes, t_{tranf} = 24-(21+1.5) =2.5 units.
If communication time between the local processor and remote processor less than 2.5 units and there are available free slots in this interval (20-24) then the task will be allocated to that processor. So processor 6 selected as a remote processor.

From the simulations, the performance of RMCT as shown Figure1, RMCT tasks give better results than RMGT



Simulations are performed using MATLAB 6.0 and complete tool under the progress. When new task arrives global scheduler automatically identifies the node for execution of new task. If it is not found suitable one then it preempts low priority task without missing deadline.

6. CONCLUSION

This paper provides a better algorithm called RMEDF for allocating execution units for both periodic and non-periodic task set for given complex real time embedded system. Also allocating periodic and non-periodic tasks to nodes.Based on these concepts we are developing a simulation tool which schedules the both periodic and non-periodic tasks. Our future works takes precedence relations and real-time faults into consideration and also effective scheduler which can suitable for any type application..

7. REFERENCES

[1] S.Cheng, J.A.Stankovic, and K.Ramamritham, Scheduling Algorithm for Hard Real Time Systems: A Brief Survey, Tutorial: *Hard Real Time Systems*, EFF Press, 1988, pp 150-173.

[2] J. D Gafford, *Rate Monotonic Scheduling, IEE Micro*, June 1991, pp 34-39 Real Time Systems by James W.S.Liu Published by Pearson Education, II Ededition, 1991.

[3] Georgio C Buttazzo, Rate Monotonic vs.EDF: Judgement Day, Real-Time Systems, 29, 5-26, 2005.

[4] Sylvain Lauzac, Rami Melhem, Fellow, IEEE, and Daniel Mosse´, Member, IEEE computer Society An Improved Rate-Monotonic Admission Control and Its Applications, IEEE Transactions On Computers, Vol. 52, No. 3, March 2003

[5] H.Chetto and M.Chetto, Some Results of the Earliest Deadline Scheduling Algorithm, *IEEE Transactions on Software Engineering* 15(10), 1989, pp 466-473

[6] Giorgio C.Buttazzo, Hard Real-Time Computing Systems Predictable Scheduling Algorithms and Applications, Kluwer Academic Publishers,1997

[7]C.Siva Ram Murthy and G.Manimaran, Resource Management in Real-Time Systems and Networks, PHI Learning Private Limited, 2009

[8] Almut Burchard, Jorge Liebeherr, Member, Yingfeng Oh, and Sang H. Son,New Strategies for Assigning RealTime Tasks toMultiprocessor Systems,IEEE Transactions On Computers, Vol.44, No.12, December 1995

A.Prashanth Rao received the Master of Technology Degree in computer science in 1999 from JNTU, Hyderabad.Presently working an Associate Professor in B.V.R.I.T, Narsapur, Medak. He is currently pursuing a Ph.D in the area real time embedded system at the Jawaharlal Nehru Technological University, Hyderabad

Dr.A.Govardhan did his BE in Computer Science and Engineering from Osmania University College of Engineering, Hyderabad in 1992, M.Tech fromJawaharlal Nehru University, Delhi in 1994 and Ph.D from Jawaharlal Nehru Technological University, Hyderabad in 2003. He is presently a Professor of CSE and Principal at Jawaharlal Nehru Technological University Hyderabad College of Engineering, Karimnagar Dt,, AP, India. He is a member on the Editorial Board of International *Journal of Emerging Technologies and Applications in Engineering Technologies and Sciences (IJ-ETA-ETS)* and International *Journal of Computer Applications in Engineering Technologies and Sciences (IJ-CA-ETS), International Journal of Advanced Computing, International Journal of Data Engineering and Computer Science,* Scientific and Technical Committee & Editorial Review Board, *World Academy of Science, Engineering and Technology.* He has 105 research publications at International/National Journals and Conferences. He is a Committee Member in *PAKDD2010, ICETCSE-2010, ICACT-2008 and NCAI06. He is Reviewer for papers of ADCOM2006, ACT2009.* He is member in various professional bodies including CSI, ISTE, FSF, IAENG and WASET. He is a member on Boards of Studies of Various Institutions including JNT University Hyderabad. He has guided 123 M.Tech projects. His areas of interest include Databases, Data Warehousing & Mining, Information Retrieval, Computer Networks, Real Time Systems, Image Processing and Object Oriented Technologies

Mr.Venugopal Cherupalli received his graduation from IETE, New Delhi and did several PGs in Electronics and Computer Science from O.U and IETE, New Delhi. He worked 22yrs in ECIL (Electronics Corporation Of India Ltd.) Also worked in US on s/w projects. Presently working in O.U.as professor.